Voor alle 6502-gebruikers die floppy-drives op hun computer willen aansluiten is deze source listing een belangrijk hulpmiddel. De floppy disk-interface van Elektuur (november en december 1982), die met goedkope standaard-onderdelen is opgebouwd, werkt met de hier beschreven software.
De object code van de gegeven programma-listing kan in een 2708-EPROM (ESS 515) worden gezet. Het disk operating system is van Ohio Scientific en heeft het typenummer OS-65D V3.1 of OS-65D V3.3. Op deze disketten staan verder een Microsoft-BASIC, een assembler, een extended monitor en diverse andere programma's.
De EPROM ESS 515 bevat diverse utility-programma's die via het hex-keyboard of door middel van een JSR-kommando kunnen worden opgeroepen:
<AD> FF17 <GO> (RUBOUT) Laden van het disk operating system van Ohio Scientific en van de Microsoft-BASIC-interpreter van de diskette naar de junior computer.
<AD> FF34 <GO> (RUBOUT) Startadres voor toekomstige uitbreidingen.
<AD> FFE2 <GO> (RUBOUT) Starten van een hulpprogramma voor het aanpassen van een OS-65D V3.1-diskette.
<AD> FFE8 <GO> (RUBOUT) Starten van een hulpprogramma voor het aanpassen van een OS-65D V3.3-diskette.
JSR FEA3 Print character-subroutine.
JSR FE1B Receive character-subroutine.

```
 -K


 ***DOS***

 COMMANDS:
 RE AS
 PUT FN
 LOAD FN

 SOURCE FILE: 4000 - 4003
 REQUIRED TRACK(S): 01

 ENTER A FILE NAME:  BOOTST

 FOR *BOOTST* IS (ARE) 12 TRACK(S) ON FLOPPY
 B*LO BOOTST

 B*RE A


 NEW?

 -N1

 -X


 ***DOS ASSEMBLER***

 THE OBJECT MODULE
 WILL BE STORED AT $E000
 DO YOU LIKE TO CHANGE IT?Y

 ENTER OBJECT START ADDRESS: $D000

 ***PASS 1***


 EDITOR?


 ***PASS 2***

 EDITOR?

 LISTING?Y
```

```
       ORG    $FC00  BOOTSTRAP LOADER
```

SOURCE LISTING OF THE BOOT STRAP LOADER
FOR ELEKTOR'S DOS JUNIOR COMPUTER OR ANY
OTHER 6502 SYSTEM

WRITTEN BY GUIDO DE CUYPER AND A. NACHTMANN

DATE: 7 SEPTEMBER 1982

THIS IS THE SOURCE LISTING OF THE MODIFIED HEX DISPLAY MONITOR OF "JUNIOR COMUTER BOOK 2" AND A BOOT STRAP LOADER FOR OHIO SCIENTIFIC SOFTWARE.

SOME ADRESSES IN THE HEX DISPLAY MONITOR HAVE BEEN CHANGED. THE BOOTSTRAP LOADER HAS GOT TWO ENTRIES: A COLD START AND A WARM START ENTRY.

THE PIA ON THE FLOPPY INTERFACE PC BOARD IS INITIALIZED BY A POWER UP AND CAN'T BE REINITIALIZED VIA THE RESET LINE!

```
***DISPLAY BUFFERS***

INL     *      $00F8
INH     *      INL    +01
POINTL  *      INH    +01
POINTH  *      POINTL +01


***TEMPS***

TEMP    *      POINTH +01
MODE    *      TEMP   +01
NIBBLE  *      MODE   +01
KEY     *      NIBBLE +01


***6532 I/O AND TIMER***


>PORT A & PORT B

PAD     *      $FA80  DATA REGISTER
```

```
PADD    *     PAD    +01 DATA DIRECTION
PBD     *     PAD    +02 DATA REGISTER
PBDD    *     PAD    +03 DATA DIRECTION


>WRITE EDGE DETECT CONTROL

EDETA   *     $FAE4  NEG EDET DISABLE PA7 IRQ
EDETB   *     $FAE5  POS EDET DISABLE PA7 IRQ
EDETC   *     $FAE6  NEG EDET ENABLE PA7 IRQ
EDETD   *     $FAE7  POS EDET ENABLE PA7 IRQ

>READ FLAG REGISTER, CLEAR TIMER & IRQ FLAG

RDFLAG *      $FAD5  BIT6=PA7 FLAG, BIT7=TIMER FLAG


>WRITE COUNT INTO TIMER, DISABLE TIMER IRQ

CNTA    *     $FAF4  CLK1T
CNTB    *     $FAF5  CLK8T
CNTC    *     $FAF6  CLK64T
CNTD    *     $FAF7  CLK1KT


>WRITE COUNT INTO TIMER, ENABLE TIMER IRQ

CNTE    *     $FAFC  CLK1T
CNTF    *     $FAFD  CLK8T
CNTG    *     $FAFE  CLK64T
CNTH    *     $FAFF  CLK1KT


***INTERRUPT VECTORS***

NMIL    *     $FA7A
NMIH    *     $FA7B
IRQL    *     $FA7E
IRQH    *     $FA7F


***RAM IN THE 6532: $FA00...$FA7F***



***6522 VIA ON JUNIOR'S INTERFACE BOARD***


> 6522 REGISTER FILE

ORB     *     $F800
ORA     *     $F801  CONTROLS HANDSHAKE
DDRB    *     $F802
DDRA    *     $F803
```

```
TALL     *      $F804  WRITE INTO LOW ORDER LATCH OF T1L-L
                       READ T1 LOW ORDER COUNTER AND(!)
                       RESET T1 INTERRUPT FLAG

TACH     *      $F805  WRITE INTO HIGH ORDER LATCH OF T1
                       WRITE INTO HIGH ORDER COUNTER OF T1
                       TRANSFER LOW ORDER LATCH INTO LOW
                       ORDER COUNTER AND RESET T1 INTERR.
                       FLAG.
                       READ T1 HIGH ORDER COUNTER

TAALL    *      $F806  WRITE INTO LOW ORDER LATCH OF T1
                       READ T1 LOW ORDER LATCH

TAALH    *      $F807  WRITE INTO HIGH ORDER LATCH OFT1 AND(!)
                       RESET INTERRUPT FLAG OF T1
                       READ T1 HIGH ORDER LATCH

TBLCL    *      $F808  WRITE INTO T2L-L
                       READ T2C-L AND CLEAR T2 INTERR.
                       FLAG.

TBCH     *      $F809  WRITE INTO T2C-H AND(!) TRANSFER T2L-L
                       TO T2C-L AND(!) CLEAR INTERR.
                       FLAG
                       READ T2C-H

SR       *      $F80A  SERIAL PORT
ACR      *      $F80B  AUXILIARY CONTROL REGISTER
PCR      *      $F80C  PERIPHERAL CONTROL REGISTER
IFR      *      $F80D  INTERRUPT FLAG REGISTER
IER      *      $F80E  INTERRUPT ENABLE REGISTER
ORAA     *      $F80F  OUTPUT REG. A, NO EFFECT ON HANDSHAKE




**********************************************
***ADDRESSES OF THE FLOPPY INTERFACE BOARD***
**********************************************


***PIA ADDRESSES (6821)***


DRA      *      $C000  DATA/DATA DIRECTION REGISTER A
CRA      *      $C001  CONTROL REGISTER A
DRB      *      $C002  DATA/DATA DIRECTION REGISTER B
CRB      *      $C003  CONTROL REGISTER B


***ACIA ADRESSES (6850)***

CACIA    *      $C010  ACIA CONTROL REGISTER
DACIA    *      $C011  ACIA DATA REGISTER
```

```
***OTHER FLOPPY I/O ADDRESSES***

STPBIT *     $FA59  AMOUNT OF STOP BITS
CNTLX  *     $FA5A  COUNT DOWN REGISTER
CNTHX  *     $FA5B
CNTHL  *     $FA5C  COUT DOWN REGISTER/2
CNTHH  *     $FA5D
TIML   *     $FA5E  CURRENT COUNT REGISTER
TIMH   *     $FA5F
TEMPA  *     $FA60
TEMPB  *     $FA61
CHA    *     $FA62
BRKT   *     $FA7C


***KERNEL ADDRESSES***

KERNEL *     $2A51  KERNEL COMMAND INTERPRETER
AHOLD  *     $2363  DOS CHARACTER STORE
MON    *     $FC00  SAVE ENTRY TO HEX DISPLAY MONITOR
LODVEC *     $00FD  LOAD VECTOR FOR THE BOOTSTRAP LOADER
KPDO   *     $2325  KEY DEPRESSED DURING OUTPUT (CTL-C)
                    SYSTEM IS BOOTED UP
SETTK  *     $26BC  MOVE THE HEAD ON THE TRACK IN ACCU
MEMHI  *     $00FF  FLOPPY LOAD POINTER
MEMLO  *     MEMHI  -01
LDHEAD *     $2754  DO A HEAD LOAD
READDK *     $2967  READ THIS TRACK INTO MEMORY POINTED
                    BY THE LOAD POINTER MEMHI,MEMLO
UNLDRD *     $2761  UNLOAD THE HEAD
SETDRV *     $29C6  SET FOR DRIVE IN ACCU
STROUT *     $2D73  PRINT THE STRING FOLLOWING THE JSR INSTR
                    THE STRING HAS TO BE TERMINATED BY
                    $00. AFTER THE STRING IS PRINTED
                    THE INSTRUCTION JUST BEHIND $00
                    IS EXECUTED!
INDST  *     $2321  INPUT DISTRIBUTOR
OUTDST *     INDST  +01 OUTPUT DISTRIBUTOR
DSTX   *     $2AC6  AN OTHER DISTRIBUTOR
SECTNM *     $265E  PRESENT SECTOR NUMBER


INVEC  *     $2301  VECTOR FOR THE PRINTER INPUT
INVECA *     INVEC  +02 2ND INPUT DEVICE
OUTVEC *     $2311  VECTOR FOR THE PRINTER OUTPUT
OUTVCA *     OUTVEC +02 2ND OUTPUT DEVICE
STRATE *     $26A3  STEP RATE DELAY

***HEX DISPLAY MONITOR***


RESET  LDAIM $1E    PB1...PB4 IS OUTPUT
       STA   PBDD
       LDAIM $01
       STA    MODE   SET ADDRESS MODE
       LDXIM  $FF
       TXS           INITIALIZE THE STACK POINTER
       SEI           DISABLE IRQ

START  CLD           DO YOUR JOB IN BINARY MODE

STARTA JSR    SCAND  SCAN DISPLAY
       BNE    STARTA WAIT UNTIL KEY IS RELEASED

STARA  JSR    SCAND  SCAN DISPLAY AND DEBOUNCE
       BEQ    STARA  IS ANY KEY DEPRESSED?
       JSR    SCAND  DEBOUNCE AGAIN
       BEQ    STARA  BRANCH ON ERROR
       JSR    GETKEY RETURN WITH KEY VALUE IN ACCU

***COMMAND FILTER***

GOEXEC CMPIM  $13    GO KEY?
       BNE    ADMODE
       LDXIM  $FF    SET THE STACK POINTER
       TXS
       CLD           JUST IN CASE
       JMI    POINTL GOTO ADDRESS POINTED BY THE DISPLAY

ADMODE CMPIM  $10    AD KEY?
       BNE    DAMODE
       LDAIM  $01    SET ADDRESS MODE
       STA    MODE
       BNE    STEPA

DAMODE CMPIM  $11    DA KEY?
       BNE    STEP
       LDAIM  $00    SET DATA MODE
       STA    MODE
       BEQ    STEPA

STEP   CMPIM  $12    + KEY?
       BNE    PCKEY
       INC    POINTL POINT=POINT+1
       BNE    STEPA
       INC    POINTH

STEPA  JMP    START

PCKEY  CMPIM  $14    PC KEY?
       BNE    ILLKEY
       JMP    GOKERN GOTO KERNEL COMMAND INTERPRETER

ILLKEY CMPIM  $15    ILLEGAL KEY
       BPL    STEPA  IF YES IGNORE IT

DATA   STA    KEY
       LDY    MODE
       BNE    ADDRES Y=0 IS DATA MODE, ELSE ADDRESS MODE
       LDAIY  POINTL GET DATA SPECIFIED BY POINT
       ASLA
       ASLA
       ASLA
       ASLA          SHIFT LOW ORDER NIBBLE TO THE LEFT
       ORA   KEY     KEY VALUE IS NOW LOW ORDER NIBBLE
       STAIY POINTL MOVE DATA TO MEMORY
       JMP   STEPA

ADDRES LDXIM $04     DO 4 SHIFTS

ADLOOP ASL   POINTL SHIFT POINTH,POINTL TO THE LEFT
       ROL   POINTH
       DEX
       BNE   ADLOOP
       LDA   POINTL
       ORA   KEY     THE KEY VALUE IS THE LOW ORDER ADDR. NIBBLE
       STA   POINTL
       JMP   STEPA


***SUBROUTINES OF THE HEX DISPLAY MONITOR***

SCAND IS A SUBROUTINE SHOWING THE DATA SPECIFIED BY
POINT.
SCANDS SHOWS THE CONTENTS OF THE DISPL. BUFFERS
POINTH,POINTL AND INH ON THE 7-SEGMENT DISPLAYS.
THE SUBROUTINE AK SCANS THE KEYBOARD. IT IS AUTO-
MATICALLY CALLED VIA SCAND OR SCANDS. AK RETURNS
WITH A=0 IF NO KEY IS DEPRESSED AND RETURNS WITH
A<>0 IF A KEY IS DEPRESSED.
WHEN SCAND OR SCANDS ARE LEFT, PORT A IS AN INPUT
PORT.


SCAND  LDYIM $00
       LDAIY POINTL GET DATA FROM MEMORY
       STA   INH     STORE DATA IN THE DATA BUFFER

SCANDS LDAIM $7F     PA0...PA6 IS OUTPUT
       STA   PADD
       LDXIM $08     ENABLE DISPLAY

SCDSA  LDA   POINTH OUTPUT POINTH
       JSR   SHOW
       LDA   POINTL OUTPUT POINTL
       JSR   SHOW
       LDA   INH     OUTPUT INH
       JSR   SHOW

SCDSB  LDAIM $00     PA0...PA7 IS INPUT
       STA   PADD

AK     LDYIM $03     SCAN 3 KEYBOARD ROWS
       LDXIM $00     RESET  ROW   COUNTER

ONEKEY LDAIM $FF
```

```
AKA    STX   PBD    OUTPUT ROW NUMBER
       INX          ENABLE FOLLOWING ROW
       INX
       AND   PAD    INPUT ROW PATTERN
       DEY          ARE ALL ROWS SCANNED?
       BNE   AKA
       LDYIM $06    TURN DISPLAY OFF
       STY   PBD
       ORAIM $80
       EORIM $FF    INVERT KEY PATTERN
       RTS
```

THE SUBROUTINE SHOW DISPLAYS THE CONTENTS OF ANY DISPLAY BUFFER ON THE 7-SEGMENT DISPLAY. THE X REGISTER IS USED AS A DISPLAY POINTER

```
SHOW   PHA          SAVE CURRENT BYTE
       STY   TEMP   SAVE Y REGISTER
       LSRA         GET HIGH ORDER NIBBLE
       LSRA
       LSRA
       LSRA
       JSR   CONVD  OUTPUT HIGH ORDER NIBBLE
       PLA          GET BYTE AGAIN
       ANDIM $0F    MASK OFF HIGH ORDER NIBBLE
       JSR   CONVD  OUTPUT LOW ORDER NIBBLE
       LDY   TEMP   RESTORE Y REGISTER
       RTS
```

THE SUBROUTINE CONVD DOES A BYTE TO 7-SEGMENT CONVERSION.

```
CONVD  TAY          USE NIBBLE AS INDEX
       LDAY  LOOK   FETCH SEGMENT PATTERN
       STA   PAD    OUTPUT SEGMENT PATTERN
       STX   PBD    OUTPUT DISPLAY ENABLE
       LDYIM $FF    DELAY A BIT

DELAY  DEY
       BNE   DELAY
       DEY          TURN THE SEGMENTS OFF
       STY   PBD
       INX          ENABLE NEXT DISPLAY
       INX
       RTS
```

GETKEY RETURNS WITH THE VALUE OF A DEPRESSED KEY IN THE ACCU REGISTER.IF AN INVALID KEY WAS DEPRESSED, IT RETURNS WITH $15 IN ACCU.

```
GETKEY LDXIM $21      START AT ROW ZERO

GETKEA LDYIM $01      GET ONE ROW
       JSR   ONEKEY
       BNE   KEYIN    A=0 THEN NO KEY DEPRESSED
       CPXIM $27      ARE ALL ROWS SCANNED?
       BNE   GETKEA
       LDAIM $15      RETURN HERE IF ILLEGAL KEY
       RTS

KEYIN  LDYIM $FF

KEYINA ASLA           SHIFT LEFT UNTIL Y=KEY NUMBER
       BCS   KEYINB
       INY
       BPL   KEYINA

KEYINB TXA
       ANDIM $0F      MASK MSD
       LSRA           DIVIDE BY 2
       TAX
       TYA
       BPL   KEYIND

KEYINC CLC
       ADCIM $07      ADD ROW DISPLACEMENT

KEYIND DEX
       BNE   KEYINC
       RTS


LOOK   =     $40      0
       =     $79      1
       =     $24      2
       =     $30      3
       =     $19      4
       =     $12      5
       =     $02      6
       =     $78      7
       =     $00      8
       =     $10      9
       =     $08      A
       =     $03      B
       =     $46      C
       =     $21      D
       =     $06      G
       =     $0E      F


*** BOOTSTRAP STARTS HERE***
```

```
DKBOOT JSR   SIGMA  INITIALIZE THE ENTIRE I/O
       JMI   LODVEC CONTINUE EXECUTION VIA THE LOAD VECTOR



***INITIALIZE THE DOS COMPUTER I/O***

SIGMA  JSR   INIDSK INITIALIZE THE 6821 PIA
       JSR   INITPR INITIALIZE THE 6532 PIA (TERMINAL)
       JSR   BOOT   LOAD TRACK 0 FROM THE DISKETTE TO $2200
       RTS


***SUBROUTINES OF THE BOOTSTRAP***

INIDSK LDYIM $00
       STY   CRA    SELECT DATA DIRECTION REGISTER A
       LDAIM $40
       STA   DRA
       LDAIM $04
       STA   CRA
       LDAIM $40
       STA   DRA
       LDXIM $04
       STX   CRA    SELECT DATA REGISTER A
       STY   CRB    SELECT DATA DIRECTION REGISTER B
       DEY          Y=FF
       STY   DRB    PORT B IS TOTAL OUTPUT
       STX   CRB    SELECT DATA REGISTER B AGAIN
       STY   DRB    SET ALL OUTPUTS HIGH
       RTS


BOOT   LDAIM $FB    FOR STEP IN DIRECTION
       BNE   TRKO    BRANCH ALWAYS

STEPLO LDAIM $02    LOAD MASK PATTERN
       BIT   DRA    ARE WE ON TRACK ZERO?
       BEQ   INDEXP IF YES, DO INDEX TEST
       LDAIM $FF    TOGGLE BITS FOR STEP OUT

TRKO   STA   DRB    STORE IN PIA
       JSR   DUMMY  DELAY SOME CYCLES
       ANDIM $F7    CLEAR STEP BIT
       STA   DRB
       JSR   DUMMY  DELAY SOME CYCLES
       ORAIM $08    SET THE STEP BIT
       STA   DRB    AND OUTPUT IT
       LDXIM $18    DELAY VALUE
       JSR   FDELAY WAIT A MOMENT
       BEQ   STEPLO TRY IT AGAIN

INDEXP LDXIM $7F    RESET HEAD LOAD BIT
       STX   DRB    HEAD LOAD
       JSR   DELAY  WAIT A MOMENT
```

```
INDBEG LDA    DRA     GET DISK STATUS
       BMI    INDBEG LOOP TILL INDEX START

INDEND LDA    DRA     GET DISK STATUS
       BPL    INDEND LOOP TILL INDEX END
       LDAIM $03
       STA    CACIA   RESET ACIA
       LDAIM $58      INITIALIZE ACIA FOR
       STA    CACIA   8 BIT EVEN PARITY, DIVIDE BY 1 RATE
       JSR    RBYTE   GET FIRST BYTE FROM DISKETTE
       STA    LODVEC +01 SAVE HIGH START ADDRESS
       TAX            SAVE IT FOR LATER USE
       JSR    RBYTE   GET SECOND BYTE
       STA    LODVEC SAVE LOW START ADDRESS
       JSR    RBYTE   READ THIRD BYTE FROM DISKETTE
       STA    $00FF   SAVE THE AMOUNT OF PAGES WE'RE GOING TO READ
                      FROM DISKETTE (PAGES/TRACK)
       LDYIM $00      INITIALIZE LOAD POINTER

RPAGE  JSR    RBYTE   READ A BYTE FROM DISKETTE
       STAIY LODVEC AND STORE IT IN MEMORY
       INY            SET UP FOR NEXT MEMORY LOCATION
       BNE    RPAGE   LOOP TILL A PAGE IS READ
       INC    LODVEC +01 ADJUST LOAD VECTOR
       DEC    $00FF   DECRMENT PAGE COUNTER
       BNE    RPAGE   LOOP TILL ALL PAGES ARE READ
       STX    LODVEC +01 RESTORE LOAD VECTOR
       LDAIM $FF
       STA    DRB     UNLOAD HEAD
       RTS


FDELAY LDYIM $F8

FDELA  DEY
       BNE    FDELA    LOOP A BIT
       EORX   $00FF   DUMMY INSTRUCTION
       DEX            ADJUST TIMER COUNT
       BNE    FDELAY LOOP TILL TIME OUT
       RTS


RBYTE  LDA    CACIA   GET ACIA STATUS
       LSRA           RECEIVER BIT IN CARRY
       BCC    RBYTE   LOOP TILL RECEIVER BUFFER IS FULL
       LDA    DACIA   GET DATA BYTE FROM ACIA

DUMMY  RTS


***INITIALIZE THE TERMINAL***

INITPR CLD
       SEI            DISABLE INTERRUPTS
       LDAIM $67
       STA    PBD
       LDAIM $00
       STA   PAD
       LDXIM $FC    YOUR EPSON LIKES THIS SPEED
       STX   CNTLX
       LDXIM $FF
       STX   CNTHX
       NOP
       LDAIM $7F
       STA   PADD   SET DATA DIRCTION
       LSRA
       STA   PBDD   SET PORT B FOR EPSON BUSY ON PB5
       LDXIM $03
       STX   STPBIT TRANSMIT NO PARITY BUT 2 STOP BITS

STRTBT BIT   PAD    WAIT FOR A START BIT
       BMI   STRTBT
       JSR   COMTIM COMPUTE THE START BIT TIME
       LSR   TIMH   DIVIDE BY 2
       ROR   TIML
       LDA   TIML
       STA   CNTHL  SAVE HALF START BIT TIME
       LDA   TIMH
       STA   CNTHH
       LDXIM $08
       JSR   DELHBT
       JSR   RECD   GET THE REST OF THE CHARACTER
       CMPIM $7F    WAS IT A RUBOUT CHARACTER?
       BNE   INITPR IF NOT, THEN TRY IT AGAIN
       RTS



***RECEIVE A CHARACTER FROM TERMINAL***

RECCHA BIT   PAD    WAIT FOR A START BIT
       BMI   RECCHA
       STX   TEMPB  SAVE X
       LDXIM $08    WE RECEIVE 8 BITS
       JSR   DELHBT DELAY FOR HALF BIT TIME

RECA   JSR   DELBIT DELAY FOR ONE BIT TIME

RECD   BIT   PAD    ONE/ZERO CHECK
       BPL   RECB   BRANCH ON ZERO
       SEC          IT IS A '1'
       ROR   CHA    ROTATE  CARRY INT CHARACTER
       DEX          SET UP FOR NEXT BIT
       BNE   RECA   ARE ALL BITS READ?
       BEQ   RECC

RECB   CLC          IT IS A '0'
       ROR   CHA
       DEX
       BNE   RECA

RECC   JSR   DELBIT WAIT FOR THE STOP BIT
```

```
       LDA    CHA     LOAD CHARACTER IN ACCU
       ANDIM  $7F     MASK OFF BIT 7
       STA    AHOLD   SAVE CHARACTER FOR DOS
       LDX    TEMPB   RESTORE X
       RTS


***COMPUTE BIT TIME***


COMTIM CLC
       LDA    CNTLX   16 BIT ADDITION
       ADCIM  $01
       STA    CNTLX
       LDA    CNTHX
       ADCIM  $00
       STA    CNTHX
       BIT    PAD     IS START BIT FINISHED?
       BPL    COMTIM
       LDA    CNTLX   SET UP FOR HALF BIT TIME COMPUTATION
       STA    TIML
       LDA    CNTHX
       STA    TIMH
       RTS



***DELAY A FULL/HALF BIT TIME***


DELHBT LDA    CNTHL   FETCH HALF BIT TIME
       STA    TIML
       LDA    CNTHH
       STA    TIMH
       SEC
       BCS    CNTDN   BRANCH ALWAYS

DELBIT LDA    CNTLX   FETCH FULL BIT TIME
       STA    TIML
       LDA    CNTHX
       STA    TIMH

CNTDN  SEC
       LDA    TIML    16 BIT SUBTRACTION
       SBCIM  $01
       STA    TIML
       LDA    TIMH
       SBCIM  $00
       STA    TIMH
       NOP
       NOP            EQUALIZE 4 MICRO SECONDS
       BCS    CNTDN   TIME OUT?
       RTS
```

```
***PRINT THE CHARACTER IN ACCU***


PRCHA  STX   TEMPA  SAVE X
       STA   CHA    SAVE CHARACTER

CTS    LDA   PBD    GET PIA DATA OF 6532
       ANDIM $40    MASK EPSON BUSY BIT
       BNE   CTS    LOOP IF PB5 IS HIGH
       LDA   PBD
       ANDIM $FE    TRANSMIT START BIT
       STA   PBD
       JSR   DELBIT DELAY ONE BIT TIME
       LDXIM $07    TRANSMIT 7 DATA BITS

PRA    LSR   CHA    SHIFT OUT CHARACTER
       BCC   PRC    BRANCH ON ZERO
       LDA   PBD
       ORAIM $01    OUTPUT A ONE
       STA   PBD

PRB    JSR   DELBIT DELAY ONE BIT TIME
       DEX          SET UP FOR NEXT BIT
       BNE   PRA    ARE ALL BITS TRANSMITTED?
       LDX   STPBIT GET AMOUNT OF STOPBITS+1

PRD    LDA   PBD
       ORAIM $01    FIRST NONE PARITY
       STA   PBD    AND THE STOP BITS
       JSR   DELBIT
       DEX
       BNE   PRD
       BIT   PAD    CHECK IF BREAK KEY IS DEPRESSED!
       BPL   BRKTST
       LDX   TEMPA  RESTORE X
       RTS

BRKTST BIT   PAD    IS THE BREAK KEY ACTIVE?
       BPL   BRKTST
       JMI   BRKT   JUMP VIA AN USER SELECTABLE VECTOR


PRC    LDA   PBD
       ANDIM $FE    OUTPUT A ZERO
       STA   PBD
       CLC
       BCC   PRB    BRANCH ALWAYS




******************************
***WARM START ENTRY FOR DOS***
```

```
*******************************

GOKERN JSR   RESTTY RESET PRINTER I/O
       JMP   KERNEL AND GOTO DOS KOMMAND INTERPRETER


RESTTY LDAIM $27
       STA   PBD
       LDAIM $00
       STA   PAD
       LDAIM $7F
       STA   PADD
       LSRA
       STA   PBDD
       RTS



*************************************
***BREAK TEST DURING BASIC OUTPUT***
*************************************

BASBRK LDAIM BASTST SET BASIC'S BREAK VECTOR
       STA   BRKT
       LDAIM BASTST /256
       STA   BRKT   +01
       LDAIM MON    IF NMI BREAK THEN GOTO HEX DISPLAY MONITOR
       STA   NMIL
       LDAIM MON    /256
       STA   NMIH
       JMP   DKBOOT NOW BOOT UP THE SYSTEM


BASTST LDAIM $03    SET BREAK FLAG (CTL-C)
       STA   KPDO   AND RETURN TO CALLER
       RTS


*****************************************
***BREAK TEST DURING DOS/AS/EM OUTPUT***
*****************************************

DOSBRK LDAIM KERNEL IF BREAK THEN GOTO KERNEL
       STA   BRKT
       LDAIM KERNEL /256
       STA   BRKT   +01
       LDAIM MON    IF NMI BREAK THEN GOTO HEX DISPLAY MONITOR
       STA   NMIL
       LDAIM MON    /256
       STA   NMIH
```

```
      JMP   DKBOOT NOW BOOT UP THE SYSTEM


***INDIRECT INTERRUPT VECTORS***

INDNMI JMI   NMIL
INDIRQ JMI   IRQL



**************************************************
*** INITIALIZE BOOTSTRAP FOR OSI MODIFICATION***
**************************************************



FIRSTX JSR   SIGMA  INIT THE DOS COMPUTER I/O
       LDAIM $28    SETP RATE DELAY
       STA   STRATE
       LDAIM $01    TRACK=1/SECTOR=1
       STA   SECTNM SAVE SECTOR NUMBER
       JSR   SETTK  MOVE HEAD ON TRACK 1
       LDAIM $2A    THE LOAD VECTOR IS $2A00
       STA   MEMHI  SAVE IT
       JSR   LDHEAD LOAD THE HEAD
       STX   MEMLO
       JSR   READDK READ TRACK 1/SECTOR 2
       LDAIM $01    SET FOR DRIVE A AND THE DISTRIBUTOR
       STA   INDST  INPUT DISTRIBUTOR
       STA   OUTDST OUTPUT DISTRIBUTOR
       STA   DSTX   COMMON DISTRIBUTOR
       JSR   SETDRV
       LDAIM RECCHA -01
       STA   INVEC  SET PRINTER INPUT VECTOR
       STA   INVECA
       LDAIM PRCHA  -01
       STA   OUTVEC SET PRINTER OUTPUT VECTOR
       STA   OUTVCA
       LDAIM RECCHA /256 HIGH ORDER ADDRESS OF RECCHA&PRCHA
       STA   INVEC  +01
       STA   INVECA +01
       STA   OUTVEC +01
       STA   OUTVCA +01
       RTS

FIRSTY INC   SECTNM SECTOR NUMBER IS 2
```

```
        LDAIM  $06    TRACK=06
        JSR    SETTK  MOVE HEAD ON TRACK 06
        JSR    READDK READ TRACK 06/SECTOR 2
        INC    SECTNM SECTNM=3
        LDAIM  $00    LOAD VECTOR =$0000
        STA    MEMLO
        STA    MEMHI
        JSR    READDK READ TRACK 06/SECTOR 3
        LDAIM  $01    SECTNM=1
        STA    SECTNM
        LDAIM  $13    TRACK=13
        JSR    SETTK  MOVE HEAD ON TRACK 13
        LDAIM  $32    SET LOAD POINTER TO $3274
        STA    MEMHI
        LDAIM  $74
        STA    MEMLO
        JSR    LDHEAD
        JSR    READDK READ TRACK 13/SECTOR 1

FIRSTZ  JSR    UNLDHD UNLOAD THE HEAD

        JSR    STROUT
        =      $0D
        =      $0A    CRLF
        =      '*
        =      'T
        =      'R
        =      'A
        =      'C
        =      'K
        =      '
        =      '0
        =      '&
        =      '1
        =      '*
        =      $0D
        =      $0A
        =      $00

        JMP    RESET  GOTO HEX DISPLAY MONITOR


*******************************
***BOOT OS 65D V3.1 FROM HERE***
*******************************


VONE    JSR    FIRSTX
        JMP    FIRSTZ
```

```
********************************
***BOOT OS-65D V3.3 FROM HERE***
********************************


VTHREE JSR   FIRSTX
       JMP   FIRSTY
```